Database Design

Project –

AirBnB

CS 6360.003

Adarsh Raghupati axh190002

Yashwanth Yella yxy180024

Ricky Mavlingcar rrm130230

# Requirements

User:

User can either be a customer or a host. (Logs in as either host or customer)

Customer:

A customer can search for properties based on different criteria to book.

A customer can book said property.

A customer may also leave feedback on the properties they have booked.

A customer can cancel the reservation

Host:

A host can have multiple properties. (Multiple property listings.)

A host receives requests for bookings.

A host receives payment from customer for completed bookings.

Property:

A property can have many features. i.e. swimming pool, bed, kitchen, parking, etc.

A property can be either housing or a restaurant.

Housing offers multiple amenities and features

Restaurant offers different amenities like wheelchair access, cab facility etc.

Booking:

A customer can have many bookings.

A booking is tied to a property and a host.

A booking can be cancelled before the check in date.

# 

# Modeling of Requirements as ER-Diagram:

A picture containing text, map

Description automatically generated

Cardinality summarized from ERD as –

1. A host can own many properties. A property can have only one owner (1:N)
2. A property can be reviewed by many customers. Each customer can review a property only once(1:N)
3. A customer can book many properties. A property can be booked by many customers at different time period(M:N)
4. A customer can search many properties. A property can be searched by many customers(M:N)
5. Each property is either housing type or a restaurant type. (1:1)
6. Each car belongs to a car-category and each car-category as many cars(1:M)
7. Each customer or host account can have only one credential (1:1)

# 

# FUNCTIONAL DEPENDENCIES

### Customer Relation:

* User\_id -> firstName, lastName, email, phone

### Host Relation:

* + User\_id -> firstName, lastName, email, phone

### Payment Relation:

* + Payment\_id -> host\_id , order\_id, payment\_type, amount

### Credential Relation:

* + user\_id -> password

### Booking Relation:

* + order\_id -> customer\_id, property\_id, host\_id, guest\_count,\_checkInDate, checkOutDate

### Property Relation:

* + property\_id -> property\_type, host\_id, capacity, price, offer,

phone, addressLine1, addressLine2, city, starte, country,

zip

* host\_id -> offer

### Availability Relation:

* + Property\_id, from\_date -> toDate

### Review Relation:

* + review\_id -> customer\_id, property\_id, rating, comments

### Housing Relation:

* + Property\_id -> wifi, pool, gym, washer, dryer, numberOfBeds, kitchen, bathRoomCount, AirCondition,parking

### Restaurant Relation:

* + Property\_id -> wifi, TV, smoking\_zone, alcohol, cabFacility, cuisines, outdoorSeating, wheelcharAccess

## Functional dependencies that violated normalization rules:

### Property Relation: Violating 3NF

* host\_id -> offer

### Property Relation: Violating 1NF

* + property\_id -> address

## Relational Schema after normalization:

A close up of a device

Description automatically generated

# SQL STATEMENTS

CREATE TABLE payment

(

  payment\_id integer,

  host\_id    integer not null,

  order\_id   integer not null,

  payment\_type VARCHAR(100),

  amount NUMERIC,

  primary key (payment\_id)

);

CREATE TABLE host

(

  user\_id    integer,

  firstname   varchar(100),

  lastname varchar(100),

  email varchar(100),

  phone varchar(12),

  primary key (user\_id)

);

CREATE TABLE customer

(

  user\_id    integer,

  firstname   varchar(100),

  lastname varchar(100),

  email varchar(100),

  phone varchar(12),

  payment\_details varchar(20),

  primary key (user\_id)

);

CREATE TABLE credential

(

  user\_id    integer,

  passwd     varchar(40),

  primary key (user\_id)

);

CREATE TABLE booking

(

  order\_id    integer,

  customer\_id   integer,

  property\_id integer,

  host\_id integer,

  guestcount integer,

  check\_in\_date date,

  check\_out\_date DATE,

  primary key (order\_id)

);

CREATE TABLE property

(

  property\_id    integer,

  property\_type   varchar(20),

  host\_id integer,

  capacity integer,

  price integer,

  phone varchar(12),

  addressline1 varchar(100),

  addressline2 varchar(100),

  city varchar(100),

  state varchar(100),

  country varchar(100),

  zip varchar(12),

  primary key (property\_id)

);

create table offers (

property\_id integer,

host\_id integer,

offer integer,

 primary key (property\_id,host\_id)

);

create table availability (

property\_id integer,

from\_date date,

to\_date date,

 primary key (property\_id,from\_date) );

create table reviews (

review\_id integer,

customer\_id integer,

property\_id integer,

rating numeric,

comments varchar(500),

 primary key (review\_id)

);

create table housing (

property\_id integer,

wifi CHAR(1),

pool CHAR(1),

gym CHAR(1),

washer CHAR(1),

dryer CHAR(1),

kitchen CHAR(1),

aircondition CHAR(1),

parking CHAR(1),

number\_of\_beds integer,

bathroom\_count integer,

 primary key (property\_id)

);

create table Restaurant (

property\_id integer,

restaurant\_name VARCHAR(255),

wifi CHAR(1),

tv CHAR(1),

smoking\_zone CHAR(1),

parking CHAR(1),

alcohol CHAR(1),

cab\_facility CHAR(1),

cuisines varchar(500),

outdoor\_seating CHAR(1),

wheelchair\_access char(1),

 primary key (property\_id)

);

ALTER TABLE Restaurant ADD CONSTRAINT fk1 FOREIGN KEY(property\_id) REFERENCES property(property\_id);

ALTER TABLE housing ADD CONSTRAINT fk2 FOREIGN KEY(property\_id) REFERENCES property(property\_id);

ALTER TABLE reviews ADD CONSTRAINT fk3 FOREIGN KEY(property\_id) REFERENCES property(property\_id);

ALTER TABLE reviews ADD CONSTRAINT fk4 FOREIGN KEY(customer\_id) REFERENCES customer(user\_id);

ALTER TABLE availability ADD CONSTRAINT fk5 FOREIGN KEY(property\_id) REFERENCES property(property\_id);

ALTER TABLE booking ADD CONSTRAINT fk6 FOREIGN KEY(customer\_id) REFERENCES customer(user\_id);

ALTER TABLE booking ADD CONSTRAINT fk7 FOREIGN KEY(property\_id) REFERENCES property(property\_id);

ALTER TABLE customer ADD CONSTRAINT fk8 FOREIGN KEY(user\_id) REFERENCES credential(user\_id);

ALTER TABLE host ADD CONSTRAINT fk9 FOREIGN KEY(user\_id) REFERENCES credential(user\_id);

ALTER TABLE property ADD CONSTRAINT fk10 FOREIGN KEY(host\_id) REFERENCES host(user\_id);

ALTER TABLE payment ADD CONSTRAINT fk11 FOREIGN KEY(host\_id) REFERENCES host(user\_id);

ALTER TABLE payment ADD CONSTRAINT fk12 FOREIGN KEY(order\_id) REFERENCES booking(order\_id);

ALTER TABLE offers ADD CONSTRAINT fk13 FOREIGN KEY(property\_id) REFERENCES property(property\_id);

# PL/SQL STATEMENTS

--Data for stored procedures

INSERT INTO CREDENTIAL VALUES (1, 'password');

INSERT INTO HOST VALUES (1, 'Ricky', 'Martin', 'rmartin@gmail.com', '6821231234');

INSERT INTO PROPERTY VALUES (1, 'restaurant', 1, 60, NULL, '6821231234', '2800 Waterview pkwy', 'Bldg 61', 'Dallas', 'TX', 'USA', '75080');

INSERT INTO RESTAURANT VALUES(1, 'Ricky Martins Bar', '1', '1', '1', '1', '1', '1', 'N/A', '0', '0');

INSERT INTO PROPERTY VALUES (2, 'restaurant', 1, 60, NULL, '6821231234', '2800 Waterview pkwy', 'Bldg 62', 'Dallas', 'TX', 'USA', '75080');

INSERT INTO RESTAURANT VALUES(2, 'Ricky Martins Fine Dining', '1', '1', '1', '1', '1', '1', 'N/A', '0', '1');

create or replace PROCEDURE is\_accessible\_restaurant AS

thisRestaurant restaurant%ROWTYPE;

CURSOR all\_accessible\_restaurants IS

SELECT \* FROM RESTAURANT WHERE wheelchair\_access LIKE '1';

BEGIN

dbms\_output.put\_line('List of accessible restaurants: ');

OPEN all\_accessible\_restaurants;

LOOP

  FETCH all\_accessible\_restaurants INTO thisRestaurant;

  EXIT WHEN (all\_accessible\_restaurants%NOTFOUND);

  dbms\_output.put\_line(thisRestaurant.restaurant\_name);

END LOOP;

CLOSE all\_accessible\_restaurants;

END;

SET SERVEROUT ON

EXECUTE is\_accessible\_restaurant;

create or replace PROCEDURE restaurant\_has\_wifi AS

thisRestaurant restaurant%ROWTYPE;

CURSOR all\_wifi\_restaurants IS

SELECT \* FROM RESTAURANT WHERE wifi LIKE '1';

BEGIN

dbms\_output.put\_line('List of restaurants with wifi: ');

OPEN all\_wifi\_restaurants;

LOOP

  FETCH all\_wifi\_restaurants INTO thisRestaurant;

  EXIT WHEN (all\_wifi\_restaurants%NOTFOUND);

  dbms\_output.put\_line(thisRestaurant.restaurant\_name);

END LOOP;

CLOSE all\_wifi\_restaurants;

END;

SET SERVEROUT ON

EXECUTE restaurant\_has\_wifi;

--data for trigger pwd\_change\_notification

insert into credentials values (2, 'pwd');

insert into customer values (2, 'Victor', 'Wooten', 'vwoot@hotmail.com', NULL);

create or replace trigger pwd\_change\_notification

 AFTER UPDATE OF PASSWD ON CREDENTIAL

 BEGIN

    dbms\_output.put\_line('password has been changed.');

 END;

-- test the trigger by updating password value

SET SERVEROUT ON

UPDATE CREDENTIAL SET PASSWD = 'pass-word' where user\_id = 2;

--data for trigger check\_customer\_payment\_details

insert into credential values(3, 'alliance');

insert into credential values(4, 'horde');

insert into host values(3, 'Anduin', 'Wrynn', 'anduin\_wrynn@battle.net',NULL);

insert into property values(3,'housing',3,75,NULL,'18002937684','Trade District',NULL,'Stormwind city',NULL,'Azeroth',NULL);

insert into housing values(3,'0','0','0','1','1','1','0','0','5','5');

insert into customer values(4,'Garrosh','Hellscream','gscream@yahoo.com',NULL,NULL);

-- trigger to check payment details of customer before they can create a booking

CREATE OR REPLACE TRIGGER check\_customer\_payment\_details

BEFORE INSERT ON BOOKING FOR EACH ROW

DECLARE

payment customer.payment\_details%TYPE;

customer\_id booking.customer\_id%TYPE;

BEGIN

    SELECT payment\_details INTO payment from customer c

    WHERE c.user\_id = :new.customer\_id;

    IF payment IS NULL THEN

        RAISE\_APPLICATION\_ERROR(-20000, 'Customer does not have payment details');

    END IF;

END;

--test trigger should raise error since the customer referenced has no payment details

insert into booking values(1,4,3,3,1,NULL,NULL);